The NIST Real-Time Control System (RCS) - A Reference Model Architecture for Computational Intelligence

James S. Albus
Chief, Intelligent Systems Division
National Institute of Standards and Technology
Gaithersburg, MD

THE NIST REAL-TIME CONTROL SYSTEM (RCS) A REFERENCE MODEL ARCHITECTURE FOR COMPUTATIONAL INTELLIGENCE

James S. Albus
Chief, Intelligent Systems Division
National Institute of Standards and Technology
Gaithersburg, MD

INTRODUCTION

The Real-time Control System (RCS) developed at NIST and elsewhere over the past two decades defines a reference model architecture for design and analysis of complex intelligent control systems. The RCS architecture consists of a hierarchically layered set of functional processing modules connected by a network of communication pathways. The primary distinguishing feature of the layers is the bandwidth of the control loops. The characteristic bandwidth of each level is determined by the spatial and temporal integration window of filters, the temporal frequency of signals and events, the spatial frequency of patterns, and the planning horizon and granularity of the planners that operate at each level. At each level, tasks are decomposed into sequential subtasks, to be performed by cooperating sets of subordinate agents. At each level, signals from sensors are filtered and correlated with spatial and temporal features that are relevant to the control function being implemented at that level (Refs. 1-6).

ELEMENTS OF INTELLIGENT SYSTEMS

The elemental functions of intelligence are Behavior Generation, Sensory Processing (or Perception), World Modeling, and Value Judgment. The remaining elements are the knowledge database and the system architecture that interconnects the functional models and databases. This is shown in Fig. 1.

BEHAVIOR GENERATION and ACTUATION
(Planning and control)

SENSING and SENSORY PROCESSING
(Filter, detect, recognize, interpret)

WORLD MODELING
(Store knowledge and predict)

VALUE JUDGMENT
(Compute cost, benefit, and uncertainty attributes)

KNOWLEDGE
(States, events, entities, attributes, maps, tasks, processes)

SYSTEM ARCHITECTURE
(Communications, timing, operating system)

Figure 1

BEHAVIOR GENERATION

Behavior generation involves both planning and control of action, as shown in Fig. 2.

In general, planning requires both spatial and temporal decomposition of tasks. Spatial decomposition is the assignment of responsibility for jobs to organizational units, or agents, and allocation of resources such as tools, materials, time, and energy. Spatial decomposition may also involve the transformation of coordinate frames in which the task is represented. For example, at the Servo level, a task is typically represented in actuator coordinates. At the Primitive Trajectory level, a task may be represented in end-effector, or tool, coordinates. At the Elementary Move level, a task may be represented in a coordinate system embedded in the surface of the object upon which the task is being performed.

Temporal decomposition requires generation of a sequence of subtasks for each agent along the time line. In many cases, the subtask sequence for the various agents must be coordinated in time and space in order to generate cooperative behavior.

Planning may be accomplished by a number of algorithms, and may be done either off-line long before action begins, or in real-time immediately before action begins. Typically, planning involves a search over the space of possible actions in order to select the best plan for execution. The purpose of planning is to find an effective and efficient path from the current state to the goal and to avoid danger along the way.

Control is the execution of the plan to produce action that is observable as behavior. Discrete control involves the sequencing of subtasks along the time line and branching on conditions. Continuous control involves moving the set point of a controller along the reference trajectory defined by the plan. Control involves detection and correction of errors.

Planning

- **Decompose tasks**
  - **-- organizationally assign responsibility & allocate resources**
  - **-- temporally develop action scenarios**
- **Search space of possible actions for "best plan"**
- **Plan paths to goals and avoid danger**
- **"Optimize" path dynamics**

Control

- **Sequence subtasks and branch on conditions**
- **Set goal points to planned trajectories**
- **Detect and correct errors**

Figure 2

SENSING AND SENSORY PROCESSING

Sensing is accomplished by a variety of sensors: visual, acoustic, tactile, smell, taste and proprioceptive. Sensors provide the input for sensory processing, or perception.

Sensory processing involves filtering input from sensory images provided by sensors. Observed images are then correlated with internal predicted images, or imagination, generated from internal knowledge sources. Both spatial and temporal correlations are important in the detection and recognition of entities and events. If observed images are correlated with internally predicted images, the input is recognized, or detected.

Recognized entities and events that fit into patterns can be clustered, or grouped, into higher-order entities and events. For example, edge points can be clustered into region boundaries. Surfaces may be grouped into objects. Phonemes may be clustered into words. Words may be grouped into sentences.

Variance between observed and predicted images can be used to update the internal knowledge database so as to estimate states, attributes, and relationships, and bring the world model into correspondence with the reality of the external world (see Fig. 3).

Filter images from sensors

Correlate with imagination

Detect or recognize entities and events

Cluster or group entities into higher order entities

Estimate states, attributes, and relationships

Figure 3

WORLD MODELING

The World Modeling function uses the results of sensory processing to update the internal knowledge database so as to keep it a current and accurate representation of the external world and of the control system itself. The World Model function answers queries from the behavior generation module, providing a best estimate of the state of the world for planning and control.

The World Modeling function also generates predictions for sensory processing and planning. Expectations of sensory input are used for correlation with observed sensory inputs. Predicted results of planned actions are sent to the Value Judgment module for evaluation. This provides the basis for planning (see Fig. 4).

Update internal knowledge database

Compute estimated state of world

Generate expected sensory input

Predict results of planned actions

Figure 4

VALUE JUDGMENT

The Value Judgment functional module computes cost/benefit evaluations of perceived events, situations, and states. It evaluates the payoff vs. risk of tentative future plans. It assigns designations of good or bad to objects, friend or foe to agents, and love or hate to persons. It computes pleasure or pain for sensory input, contentment or anger for the current internal state, and hope or fear for imagined future conditions. The Value Judgment function also assesses the level of confidence or uncertainty to be ascribed to sensory observations or internally generated imaginations as shown in Fig. 5.

Value judgment := An evaluation of current reality past history, or future expectations. An estimate of cost, risk, or benefit.

Value judgments compute state-variables that may be assigned to objects, events, persons, or regions of space.

Examples of value state-variables :=
benefit-cost, payoff-risk, good-bad, pleasure-pain, hope-fear, love-hate, friend-foe, contentment-anger, confidence-uncertainty

Figure 5

KNOWLEDGE REPRESENTATION

Knowledge is represented in a distributed knowledge database in a number of forms, as shown in Fig. 6.

Estimates of the state of objects and systems in the world are represented by state vectors.

Attributes of entities and the relationships between entities are represented in symbolic lists that can be indexed by name. Entity representations are defined by symbols and strings, organized in frames, lists, and graphs. Relationships are defined by pointers.

Maps and images are represented by arrays of attribute-value pairs that are indexed by their position in the array. There typically exists a topographical mapping between the location of the sensor (such as a photo-receptor in the retina, or a touch sensor on the finger-tip) and the location of the attribute-value pair in the array (such as the visual or somatosensory cortex).

Additional types of knowledge are:

Task knowledge which embodies skills in how to do things;

Laws of physics which embody models of how things are expected to behave in the world; and

Laws of math and logic which embody concepts of causality, and provide the basis for deductive and inductive reasoning.

States (conditions at time t)

Entities
- **Symbolic (symbols and strings)**
- **Indexed by name**
- **Lists of attribute-value pairs**

Maps
- **Iconic (images)**
- **Indexed by position**
- **Arrays of attribute-value pairs**

Task knowledge -- skills in how to do things
Laws of physics -- models of how things behave
Laws of math -- models of how things relate

Figure 6

SYSTEM ARCHITECTURE

The elements of intelligence are organized as shown in Fig. 7 such that the Behavior Generating module can submit tentative plans to the World Model module, which generates expected results that are sent to the Value Judgment module which evaluates the cost/benefit and risk/payoff. This is returned to the Behavior Generating module which decides either to execute the plan or to generate an alternative plan. A series of tentative plans can be submitted for simulation and evaluation before an acceptable plan is chosen for execution. The loop between the Behavior Generation, World Modeling, and Value Judgment modules thus constitutes a planning loop. The World Model also provides estimated state information to the Behavior Generating module for its planning and control functions. Output from the Behavior Generating module represents commanded actions to be carried out by actuators that act on the external world environment.

Sensors sense events in the environment and provide observed input to Sensory Processing (or Perception) modules. The World Model generates predicted sensory input based on the estimated state of the environment. The Sensory Processing modules filter the observed input, and apply correlation and difference operators to the observed and predicted inputs. Strong correlations indicate recognition or detection. Variance between observations and predictions provide the information necessary to update the Knowledge Database. The Sensory Processing modules also group or cluster recognized entities into higher-level entities, situations, and relationships. Perceived situations are analyzed by the Value Judgment module as to whether they are good or bad, and are labeled as such in the Knowledge Database.

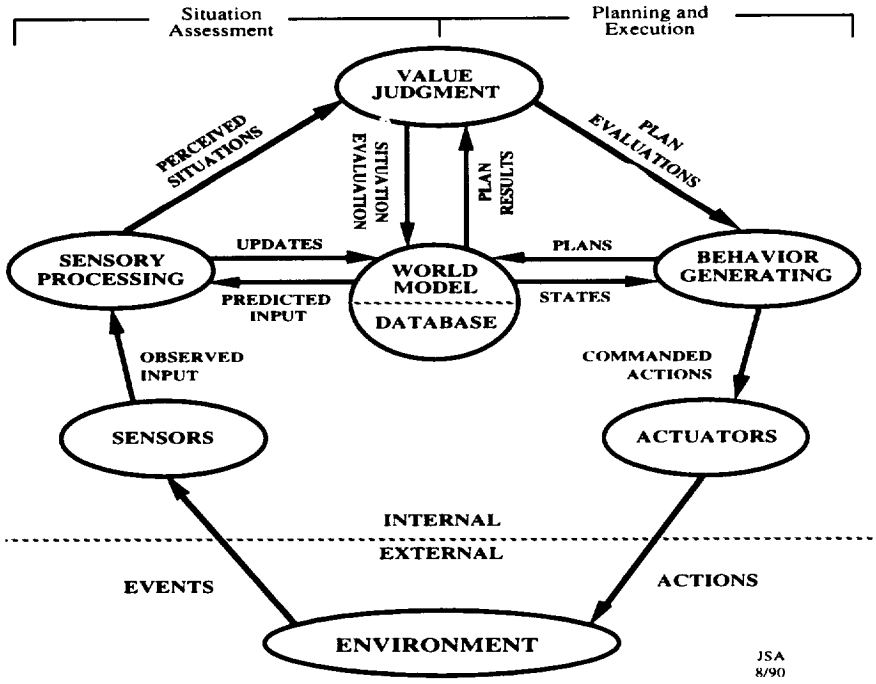

Figure 7

ORGANIZATION OF INTELLIGENT SYSTEMS

A more abstract model of the functional elements of intelligent systems, organized into computational nodes and interconnected to each other and to the knowledge database by a communications system, is shown in Fig. 8. The Behavior Generating element receives goals as input and generates action as output. The Sensory Processing element receives observed input from sensors and generates reports as output. Sensory input that reports success in achieving goals is rewarding. Failure to achieve goals is punishing. Rewarding and punishing sensory feedback can be used for learning.

Learning may change algorithms in the Behavior Generating, Value Judgment, or Sensory Processing systems. Memory is a form of learning that changes what is stored in the Knowledge Database.

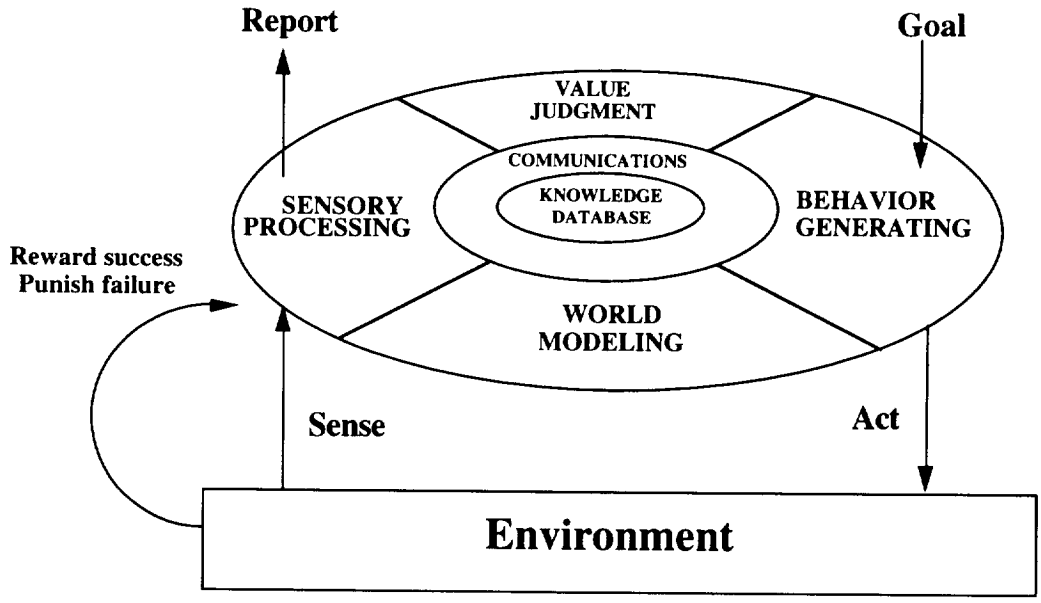

Figure 8

HIERARCHICAL ORGANIZATION

Complexity can be managed through hierarchical layering of computational nodes. Higher level Behavior Generating modules have longer range goals and make plans with longer planning horizons and less detail. At each hierarchical level, Behavior Generating modules decompose longer range goals into strings of shorter range subgoals with increasing detail and more immediate reactivity to sensory feedback, as shown in Fig. 9.

Higher-level Sensory Processing modules have longer temporal intervals over which historical trends are integrated and wider spatial windows over which entity attributes are averaged. Knowledge Databases at higher levels contain more general properties of larger-scale entities and more general relationships between groups of entities.

Reactive feedback loops are closed at every level, with higher bandwidth loops at lower levels, and slower more deliberative reactions at higher levels.

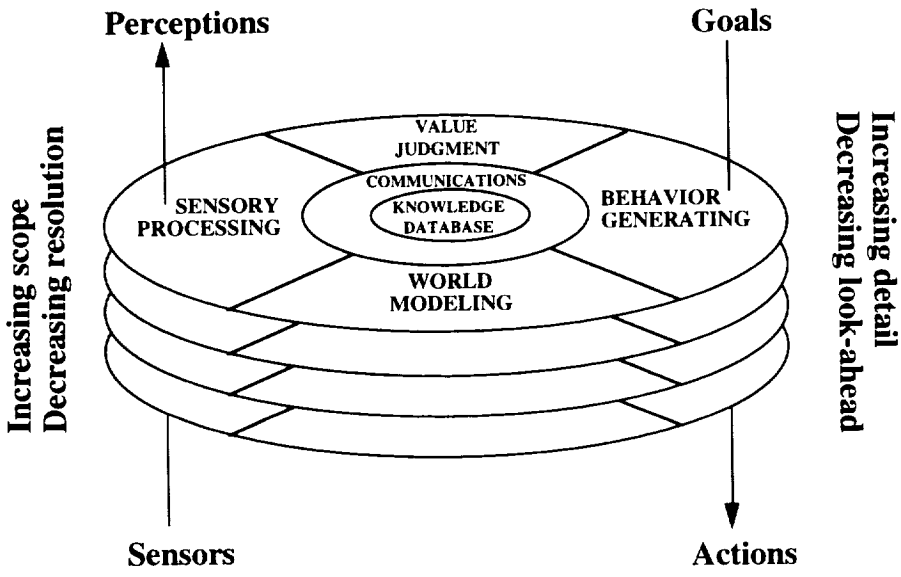

Figure 9

SYSTEM ARCHITECTURE

The NIST Real-time Control System (RCS) is a Reference Model Architecture for Intelligent Systems, as shown in Fig. 10. Processing nodes consist of Behavior Generating (BG), Sensory Processing (SP), World Modeling (WM), and Value Judgment (VJ) (hidden behind WM) modules. These are organized such that the BG modules form a command tree. Information in the Knowledge Database (KB) (contained in WM) is shared between WM modules in nodes within the same subtree. Relational links exist between entities in the KB at different levels. On the right, are examples of the functional characteristics of the BG modules at each level. For example, the planning horizon and typical output is listed under the name of each level. On the left are examples of the type of entities recognized by the SP modules and stored by the WM in the Knowledge Database at each level. Sensory data paths flowing up the hierarchy form a graph, not a tree.

This diagram is for an individual machine system. This system is decomposed into subsystems of Attention, Communication, Locomotion, and Manipulation. Each of these subsystems is further decomposed until at the bottom level , there are actuators and sensors. Above the individual level, there are control levels that deal with other individuals in a group, between the self group and other groups, etc.

The NIST RCS (Real-time Control System)
Reference Model Architecture for Intelligent Systems

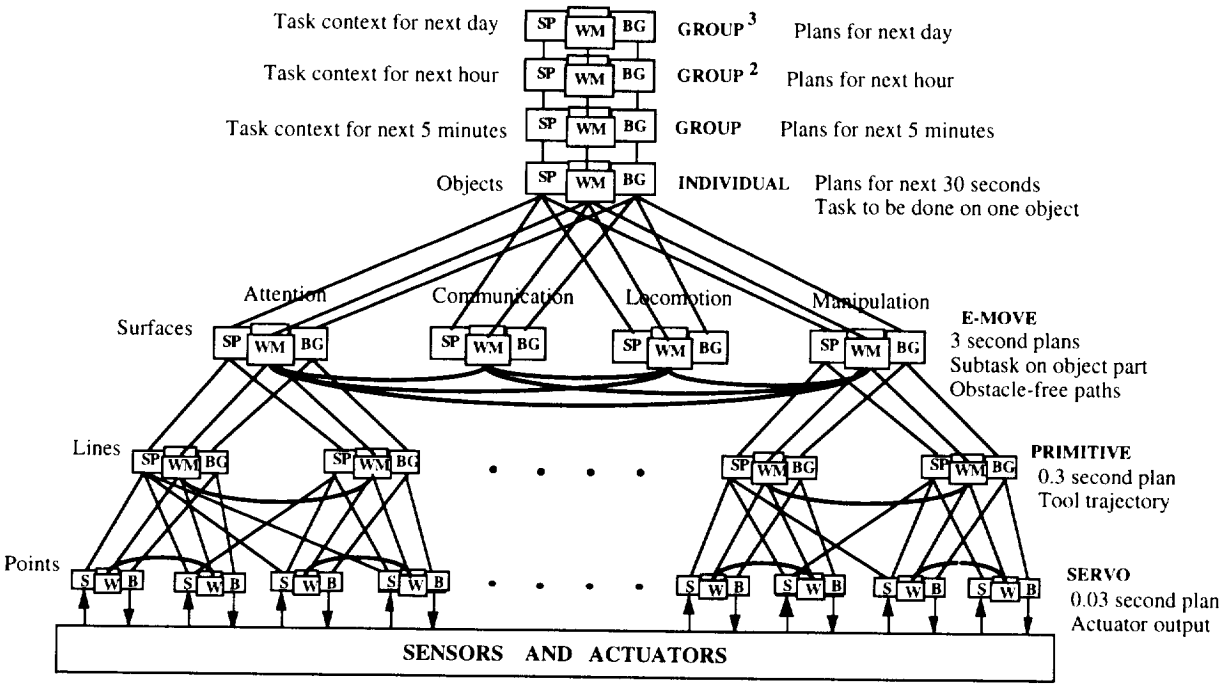

Figure 10

RELATIONSHIPS WITHIN A RCS NODE

The Behavior Generating (BG) modules contain Job Assignment (JA), Scheduling (SC), Plan Selector (PS) and Executor (EX) submodules, as shown in Fig. 11. The World Modeling (WM) module contains a plan simulator and mechanisms for updating the Knowledge Database (KD), which contains both long term and short term symbolic representations and short term iconic images. The Sensory Processing (SP) module contains filtering, detecting, and estimating algorithms, plus mechanisms for comparing predictions generated by the WM module with observations from sensors. It has algorithms for recognizing entities and clustering entities into higher-level entities. The Value Judgment (VJ) module evaluates plans and computes confidence factors based on the variance between observed and predicted sensory input.

Task commands are input to the BG module from an EX submodule at the next higher level in the RCS hierarchy. The task commands are of the form

Do <Action> On <Object> To Achieve Goal <x*>

The <Action> consists of a verb that specifies the name of the task to be performed, plus parameters, such as priority, or speed, or timing requirements, that specify how the task is to be performed. The <Object> is a noun, the object of the action verb. The <Object> specifies the thing, or things, to be acted upon by the task. The goal <x*> defines the state to be achieved or maintained by the task.

The Job Assignment (JA) submodule decomposes the task specified by <Action> into jobs to be performed by subagents and assigns resources to the agents. JA may also transform the coordinate frame in which actions are expressed. For each subagent, there is a Scheduler (SC) submodule which schedules a sequence of subtasks that accomplish the job assigned to its agent, and coordinates its schedule with other subagents. The output of the SC submodules is a tentative plan that is sent to a simulator in the WM module. The expected results generated by the WM simulator is forwarded to the VJ module for cost/benefit/risk/payoff analysis. The VJ evaluation is returned to the Plan Selector (PS) submodule which decides whether to replan in an attempt to find a more desirable plan, or to send the best of the plans generated so far to the Selected Plan register for execution. The selected plan defines a path from the current state $\hat{x}$ to the goal state x^*. This path is a reference trajectory x^{**} that becomes input to the Executor submodules for each of the subagents. $\hat{x}$ is the WM best estimate of the state of the world. $\hat{x}$ provides feedback to be compared with the reference trajectory x^{**}. Errors between $\hat{x}$ and x^{**} are then used to compute actions designed to null the difference between the reference trajectory and the state of the world.

<Object> provides input to the WM to access the long term memory section of the Knowledge Database. Information stored in long term memory about the <Object> is transferred into short term working memory, so that the <Object> becomes an object of attention. Attributes and state of the <Object> provide input to a graphics engine to generate masks and windows for filtering, and a template for correlation matching with sensory observations. Comparisons between sensory observations and world model predictions provide the information to update the attributes and state of the <Object> in the WM so that the control system can manipulate the <Object> in a way designed to achieve the goal state x^*.

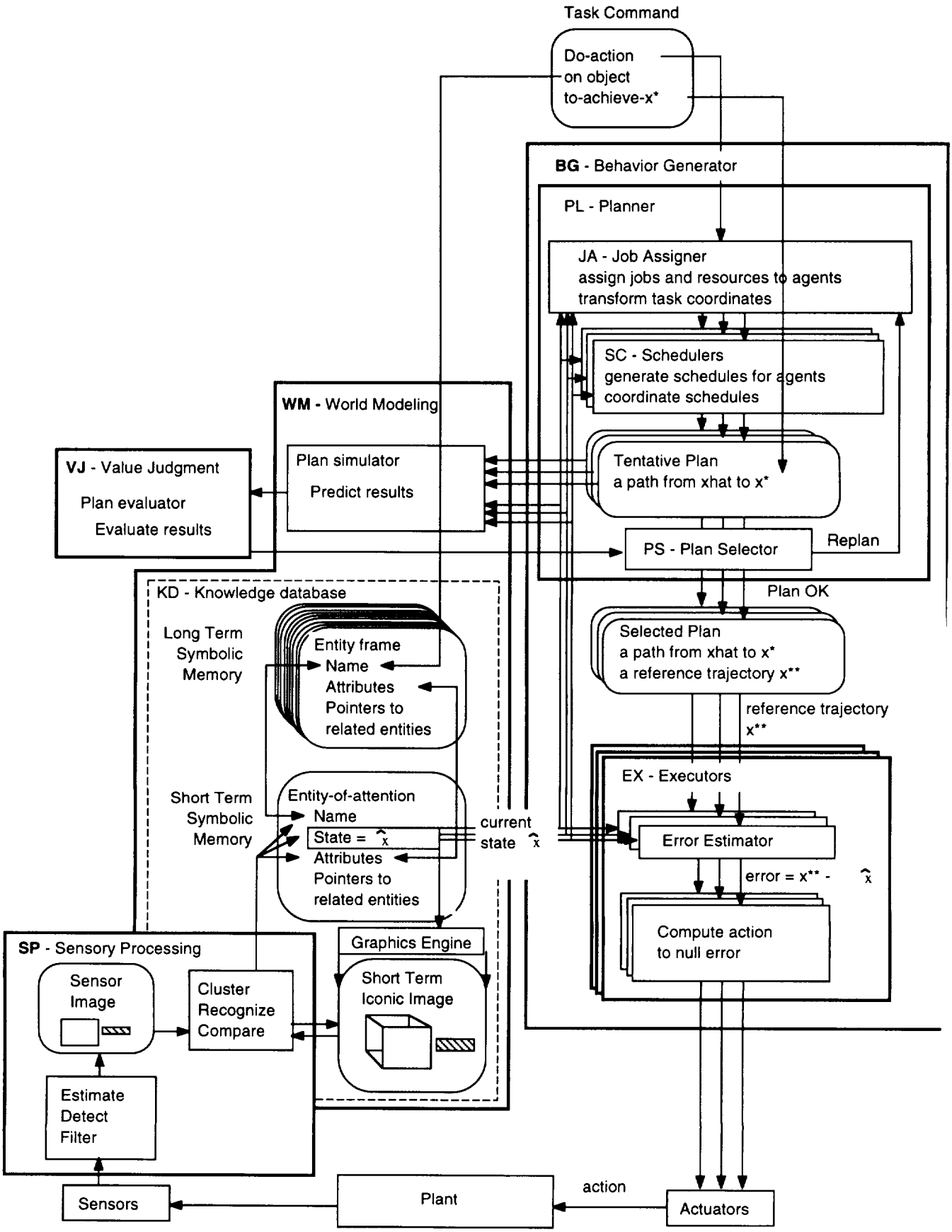
Task Command
Do-action
on object
to-achieve-x*
BG - Behavior Generator
PL - Planner
JA - Job Assigner
assign jobs and resources to agents
transform task coordinates
SC - Schedulers
generate schedules for agents
coordinate schedules
WM - World Modeling
VJ - Value Judgment
Plan evaluator
Evaluate results
Plan simulator
Predict results
Tentative Plan
a path from xhat to x*
PS - Plan Selector
Replan
Plan OK
KD - Knowledge database
Long Term
Symbolic
Memory
Entity frame
Name
Attributes
Pointers to
related entities
Selected Plan
a path from xhat to x*
a reference trajectory x**
reference trajectory
x**
EX - Executors
Short Term
Symbolic
Memory
Entity-of-attention
Name
State = x̂
Attributes
Pointers to
related entities
current
state x̂
Error Estimator
error = x** - x̂
Compute action
to null error
SP - Sensory Processing
Sensor
Image
Cluster
Recognize
Compare
Graphics Engine
Short Term
Iconic Image
Estimate
Detect
Filter
Sensors
Plant
action
Actuators

Figure 11

OPERATOR INTERFACES

An operational RCS system provides operator interfaces to each of the modules in each of the nodes. These allow the operator to view plans, including part drawings with tolerances and bill of materials, as well as assembly drawings with exploded views, and diagrams of planned tool paths. The operator may also display sensory data that show measurements made by inspection instruments, or knowledge database diagrams showing layouts of work sites. Displays may include symbolic and numeric data, dials and gauges, and plant diagrams showing state of control modules and flow of parts and material.

Operator interfaces also provide means for inputting operator commands such as feed-rate override, jog, feed-hold, pause, and emergency stop. Input devices may include joysticks, keyboards, function keys, and voice input (see Fig. 12).

Display plans
- **Part drawings with tolerances and materials**
- **Assembly drawings with exploded views**
- **Tool path diagrams**

Display sensory input
- **Site metrology**
- **Inspection measurements**

Display world model knowledge
- **Symbolic, dials and gauges, state diagrams**
- **Maps with feature overlays**

Input commands, queries
- **Joystick, keyboard, function keys, voice**

Figure 12

ENGINEERING METHODOLOGY

The design of an intelligent machine system, outlined in Fig. 13, begins with a scenario analysis wherein the requirements, objectives, and operational behavior of the system are specified in great detail. Experts in the desired performance are interviewed and specifications are developed for both normal operating conditions and error recovery procedures. This is followed by a definition of the hierarchy of task vocabularies required at each level to achieve the desired performance, plus a specification of the task knowledge needed to accomplish each of the tasks in the vocabulary. It is then possible to specify the world model knowledge necessary to support the task decomposition processes, and define the entities, events, agents, attributes, maps, lists, and state variables needed for world model representation. Next the sensors and sensory processing algorithms with the capability to detect events, recognize features, and provide the required world model data are defined. Then the set of messages and communication protocols is defined that can support the communication of task commands, status reports, world model queries and responses, and data flow in the sensory processing hierarchy. At each level, timing, sampling, and spatial/temporal resolution must be defined. Finally, the computing platforms, memory requirements, and communication protocols can be specified, and operating systems and operator interfaces can be defined (Refs. 7 and 8).

This methodology typically needs to be integrated through a number of cycles before the system design stabilizes.

Scenario analysis
Task decomposition and task knowledge
World model knowledge representation
Maps, lists, and state variables
Entities, events, agents, states, and attributes
Sensory processing algorithms and filters
Event detection and feature recognition
Communication messages and protocols
Timing, sampling, and spatial resolution
Computing platforms, memory, buses, and LANs
Virtual machines
Operating systems
Operator interfaces

Figure 13

SOFTWARE DEVELOPMENT TOOLS

Intelligent systems are typically too complex to be designed and developed using conventional programming practices. Advanced software development tools, such as suggested in Fig. 14, are required in order to achieve any degree of proficiency in intelligent system design. Below are some of the types of tools and representations that are needed.

Task frames
Message formats
Product data representations
Object oriented analysis
Functional module definition
Data flow analysis
Simulation tools
System configuration tools
Frequency response and transient analysis plots
State diagrams and Petri nets
Path planners and motion control libraries
Programmable logic analysis

Figure 14

REFERENCES

1. Albus, J. S., "Outline for a Theory of Intelligence," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 21, No. 3, May/June 1991, pp. 473-509.

2. Albus, J. S., *Brains, Behavior, and Robotics*, Byte/McGraw-Hill, 1981

3. Albus, J. S., "A Reference Model Architecture for Intelligent Systems Design," in *An Introduction to Intelligent and Autonomous Control*, Antsaklis, P. J. and Passino, K. M. (eds.), 1993.

4. Albus, J. S., McLean, C. R., Barbera, A. J. and Fitzgerald, M. L., "Architecture for Real-Time Sensory-Interactive Control of Robots in a Manufacturing Facility," *Proceedings of the Fourth IFAC/IFIP Symposium -- Information Control Problems in Manufacturing Technology*, Gaithersburg, MD, Oct. 26-28, 1982.

5. Albus, J. S., McCain, H. G. and Lumia, R., *NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)*, NISTTN 1235, 1989 edition, National Institute of Standards and Technology, Gaithersburg, MD, April 1989 (supersedes NBS Technical Note 1235, July 1987).

6. Senehi, M. K., Kramer, T. J., Michaloski, J., Quintero, R., Ray, S. R., Rippey, W. G. and Wallace, S., *Reference Architecture for Machine Control Systems Integration: Interim Report*, NISTIR 5517, National Institute of Standards and Technology, Gaithersburg, MD, 1994.

7. Huang, H. M., Quintero, R. and Albus, J. S., "A Reference Model, Design Approach, and Development Illustration Toward Hierarchical Real-Time System Control for Coal Mining Operations," *Advances in Control and Dynamic Systems*, Academic Press, July 1991.

8. Quintero, R. and Barbera, A. J., "A Software Template Approach to Building Complex Large-Scale Intelligent Control Systems," *Proceeding of the 8th IEEE International Symposium on Intelligent Control*, Chicago, IL, Sept. 25-27, 1993.